RMIT University
Programming 1
COSC1073 / 2362

Lecturer

- Daryl D'Souza (daryl.dsouza@rmit.edu.au)
- Office location: 14.09.18

Head Tutor

- Craig Hamilton (craig@cs.rmit.edu.au)

  Contact him for any Admin matters (coursework, absence, etc)
- Office location: 10.9.37C

COSC1072/2362 Overview

1. Introduction, programming languages, OO
2. Variables, identifiers, Operators and precedence.
3. Decisions, Algorithms and Stepwise refinements.
4. Repetitions, Intro. to Arrays.
5. Writing simple classes
6. Arrays, methods and parameters, Applets
7. Inheritance and Polymorphism
8. More modifiers: static and final.
9. Arrays Strings revisited.
10. Exception handling
11. Java Collection Framework
12. File manipulation
13. Revision

Programming 1

- One of the most important subject in CS as it is prerequisite for many others.
- Practical subject - requires at least 8 -10 hours of programming practice per week.
- Assessment 1. 50% Exam

  2. 50% Coursework

  Programming Assignments 25%
  (3 to be ticked off, 3 to be submitted)

  1 WebLearn Tests 5%

  Mid-semester Test 10%

  Tutorials (10 x 1) 10%

Note: You need to pass both components (1 and 2).

Resources

Prescribed:

- Programming 1 *Student Notes (2007)*, Department of Computer Science, RMIT
- Daniel Liang, *Introduction to Java Programming*, Sixth Edition, Prentice Hall, ISBN 0-13-222158-6

Recommended Reference:
Absolute Java, by Walter Savitch
Java Concepts, by Cay HorstMann
There are numerous texts in Java
On-line Materials

- WebLearn Quiz
- Online Materials (Exercise, Review Questions)
- Copy of these Slides, detailed notes

Assessment Schedule

For detailed up to date guide refer to Course Guide

Week 2 – Week 11	
(1 Mark for completing Review Questions before Tutes)	10%
Week 4 –	
First assignment must be ticked off during the lab hour	(2%) (4%)
Week 6–	
Second Assignment submit by Monday	(5%) (5%)
Week 8–	
Third assignment must be ticked during the lab hour	(4%) (5%)
Mid Semester Test to be taken during the week	(10%).
Week 10 -	
Fourth assignment submit by Monday	(4%) (5%)
Week 11 -	
WebLearn Test to be taken in the lab hour	(5%).
Week 12 -	
Fifth assignment must be ticked off during the lab hour	(2%) (5%)
Sixth assignment submit by Friday	(8%) (11%)
Week 14	
Exam	(50%)

Recipe for Doing Well

- Read the Subject Notes, on-line materials before going for Lectures. Review it afterwards.
- Attend all Lectures (3x1hr), Labs (2hr) & Tutes (1hr).
- Attempt the review questions before tutes (10%)
- Start Assignments Early - always takes longer than you think. They are all made available now.
- Complete all the lab exercises even if you’re not required to submit them.
- Read the relevant section in text book before lecture
- Start Writing Programs in week 1 itself !!! Those who start early always do well.

Week 1

- Introduction to the course
- Origin of Java
- Types of Computer Languages
- Compiling and Interpreting
- Applets vs Applications
- Components of a Java Program
- Classes and Objects
- Using the Java Docs to view the Class details
- Creating and Manipulating objects
- Packages in Java
- Algorithms
- Programming Style
- Programming Errors

Read Pages 4 - 25

Origins of Java

- Originated as part of a Research Project
- Embedding networking capabilities and Cross platform portability
- Originally called Green, then changed to Oak, finally Java
- Design decisions drew from C++, Eiffel, SmallTalk
- Launched as an Internet extension for browser early 1995.

Java's Design Philosophy

- Simple
- Familiar to Developers
- Object Oriented
- Portable
- Multi-threaded
- Robust and Secure

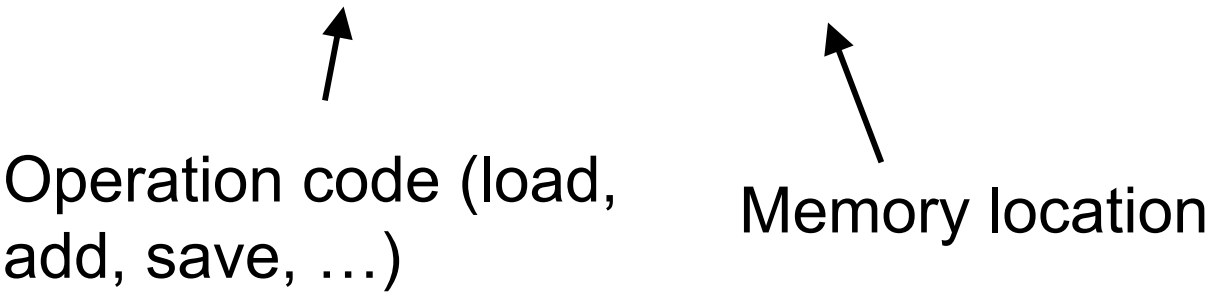
What are the programming skills currently most sought after by employers? _____

Machine language

Languages that a computer can directly understand.
Differs from Machine to Machine.

Example:

```
0010 0000   1100 0100
0011 0111   1100 0100
```



Assembly language

- One level higher than machine languages
- Specified in terms of basic operations and symbolic names.

```
LOAD    Hcost
MOVE    Tax
ADD     Profit
```

High- Level languages

- Programs specified in an English-like syntax, and a compiler is used to translate it.

```
for (int year = 2000; year <2010; year++)
    System.out.println("Year is " + year);
```

- Each compiler produces a translation for a specific computer.
- Same source program can be run in another computer if it is compiled by the appropriate compiler.

Divisions of High- Level Languages

- Procedural - C, Pascal

  divided into procedures
- Object-oriented - Java, C++, SmallTalk

  based on interaction of objects
- Logic languages - prolog

  setting up goals

Traditional Steps:
Compiling, Linking and Executing

Standard way of producing and creating an application

1. Create the program.

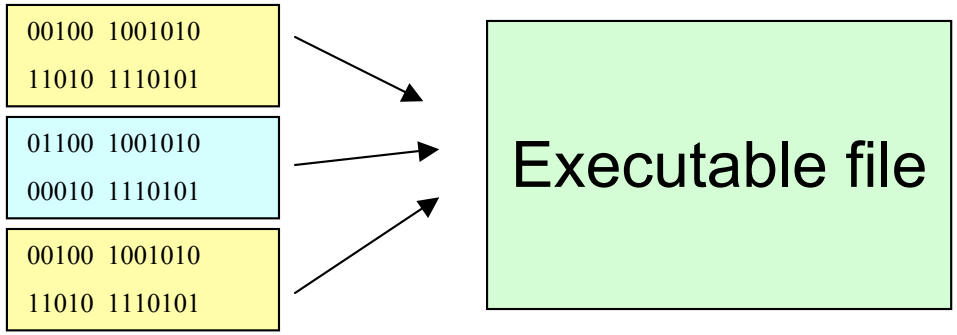
```
Int add(int x, int y)
{ int z = ...
```

2. Traditional compilers translate source code into machine language directly.

```
Int add(int x, int y)
{ int z = ...
```

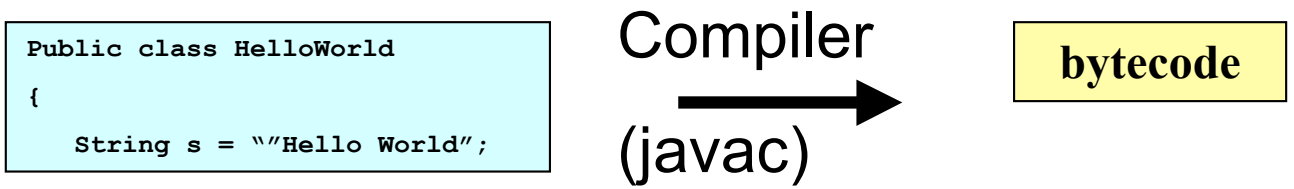
```
01100  1001010
00010  1110101
```

3. Linkers link Standard code and library modules into executables

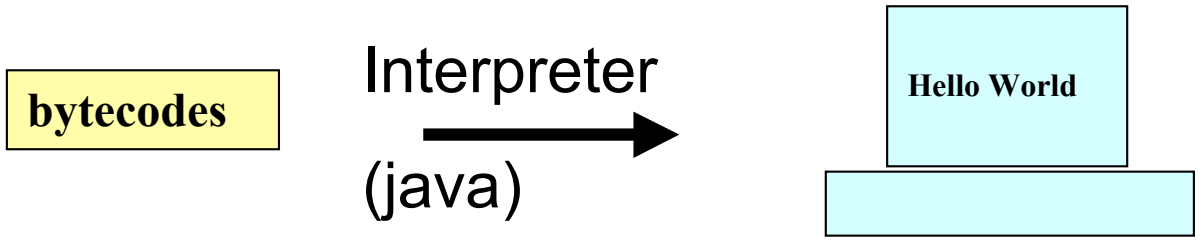


What's different about Java ?

- Java compilers translate source code into an intermediate language called (platform independent) bytecode.



- The Java interpreter reads the bytecodes (loads the necessary library bytecodes) and executes it on a specific machine.



- Unlike machine code Java bytecode is not tied to any particular machine making it architecture neutral.

Editing, Compiling & Interpreting

Step1 - Create the code using an editor

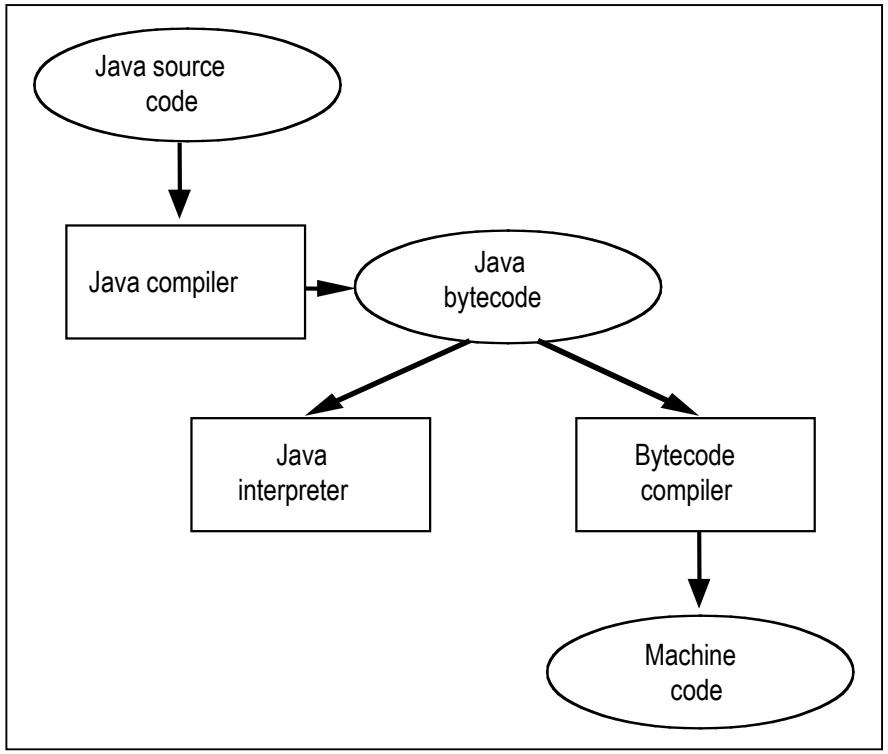
```
H:\>edit HelloWorld.java
```

Step2 - Compile the code using javac

```
H:\>javac
HelloWorld.java
```

Step3 - Interpret and execute the program

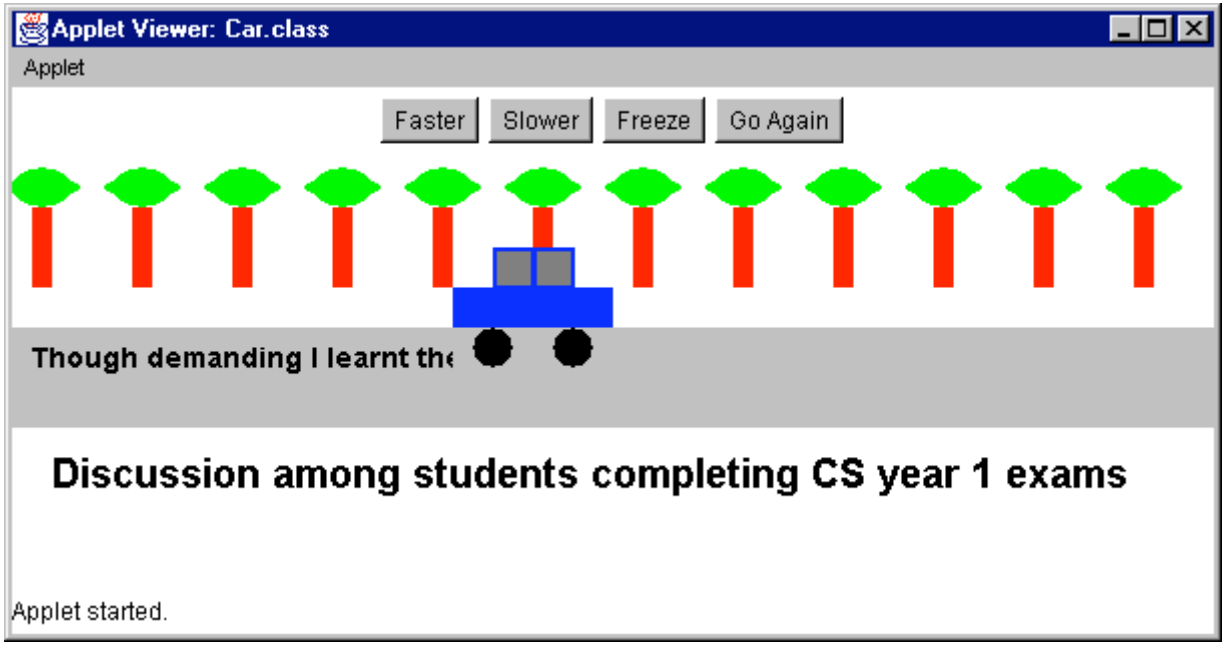
```
H:\>java HelloWorld
```



Give one benefit and one drawback of Java over other high level languages such as C or Pascal.

Two type of Java Programs: Applets

- Designed to be remotely downloaded as part of an HTML page and executed in the client machine.
- They are executed in a browser or appletviewer.



Two type of Java Programs: Stand Alone Applications

- Stand-alone applications are similar to other applications written in high level language.
- Though stand-alone applications can incorporate Graphical user interface and events (mouse-moved, button-pressed) we will concentrate on console based applications in this module.

```
Enter Name   Robert Homewood
Enter Course       CS
Hello Robert Homewood Welcome to CS Course
```

Console application

First Java Program

```
// The first Java program
// The famous Hello World!
public class HelloWorld  {
    static int year;
    public static void main(String[] args){
       year = 2007;
       System.out.println(new String("Hello world ") +
year);
    }
}
```

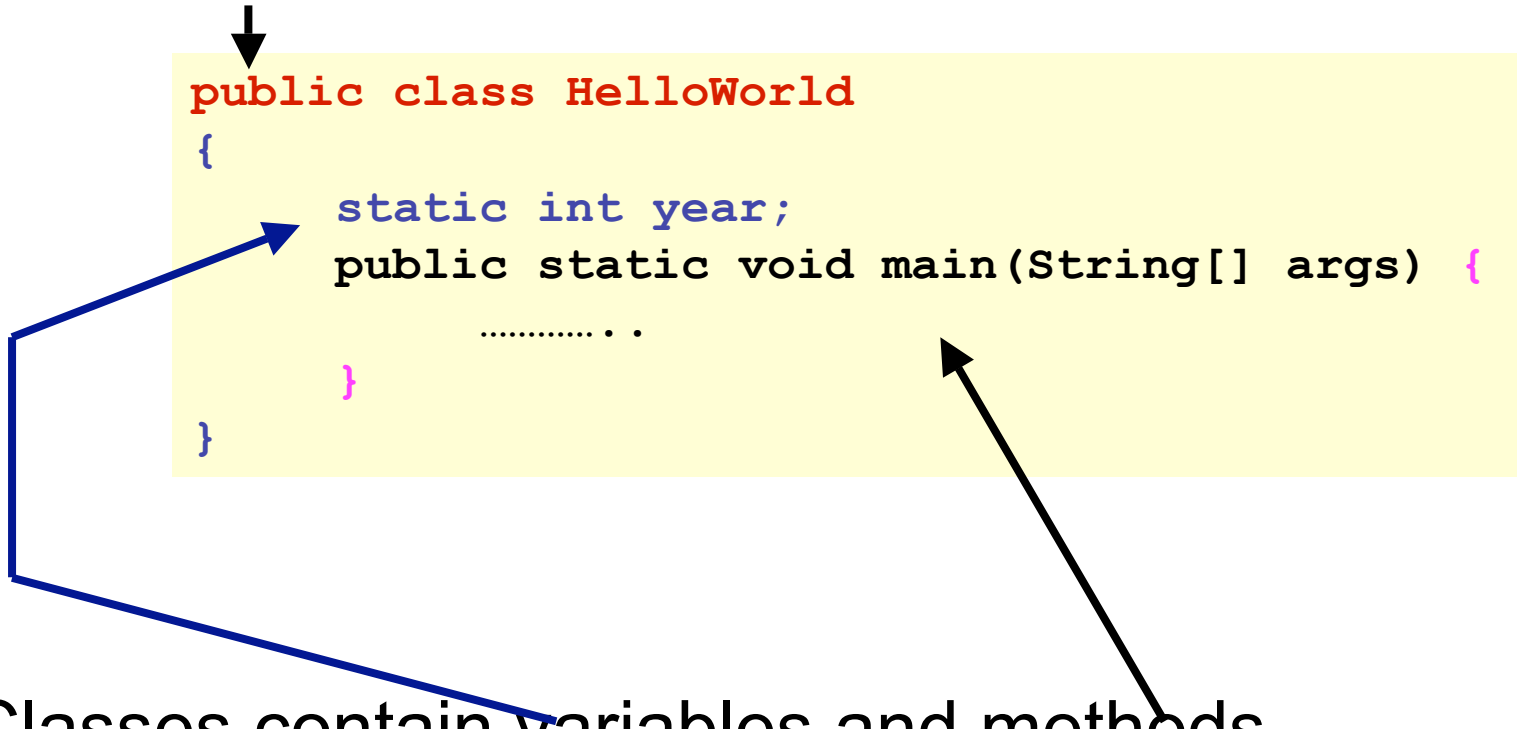
H:\>javac HelloWorld.java

H:\> java HelloWorld

H:\>Hello World 2007

Java is about creating & using classes !!!

All Java programs must have at least one of this



Classes, methods and related statements are enclosed between { ... }

Making code more Readable

- Use Comments, they make your programs readable
- They can appear anywhere in a program
- They are ignored by compiler
- Type 1 // up to the end of the line is a comment
- Type 2 /* all character enclosed between these
  are comments - hence ignored */
- Don't overdo it x = y; // assigning y to x (obvious!)

```
/* My first Java program prints the famous Hello World!
 The tradition says all great programmers must do this*/
public class HelloWorld  {
     static int year;      // current year
     public static void main(String[] args){
          ……..
     }
}
```

Quiz Time:What is the output ?

```
// The first Java program
// The famous Hello World!
public class HelloWorld  {
    static int year;
    public static void main(String[] args){
       //   year = 2001;
       System.out.println("Hello world " + year);
    }
}
```

(a) Hello World

(b) Hello World 0

(c) Hello World 2001

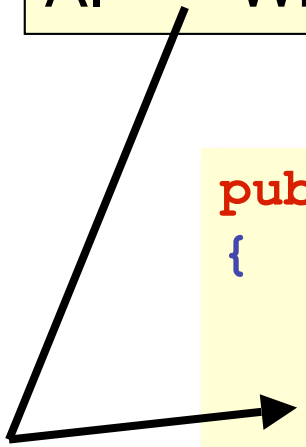
Where does the execution begin ?

Recall how we ran the program HelloWorld with:

H:\> java HelloWorld

Q. Where does the execution begin ?

A. With the main() method of HelloWorld



```
public class HelloWorld
{
    int year;
    public static void main(String[] args)
    {
            ……..
    }
}
```

What will be the output when we compile and interpret the code below ?

```
public class HelloThere
{
   public static void anotherMethod( )
   {
      System.out.print("There ");
   }
   public static void main(String[] args)
   {
      System.out.print("Hello ");
   }
}
```

(A) Hello

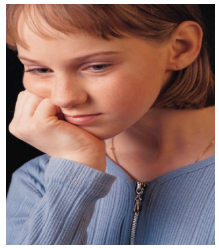
(B) There

(C) Hello There

(D) There Hello

What will be the output now ?
(note the additional line in main())

```
public class HelloThere
{
    public static void anotherMethod( )
    {
        System.out.print("There ");
    }
    public static void main(String[] args)
    {
        System.out.print("Hello ");
        anotherMethod();
    }
}
```



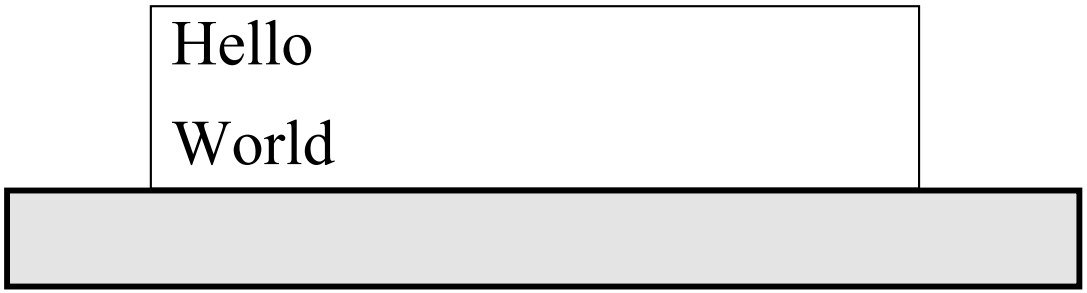
(A) Hello
(B) There
(C) Hello There
(D) There Hello

Why do we use methods?

Terminating statements

- Each statement must ends with a semicolon ';'

```
System.out.println("Hello");
System.out.println("World");
```

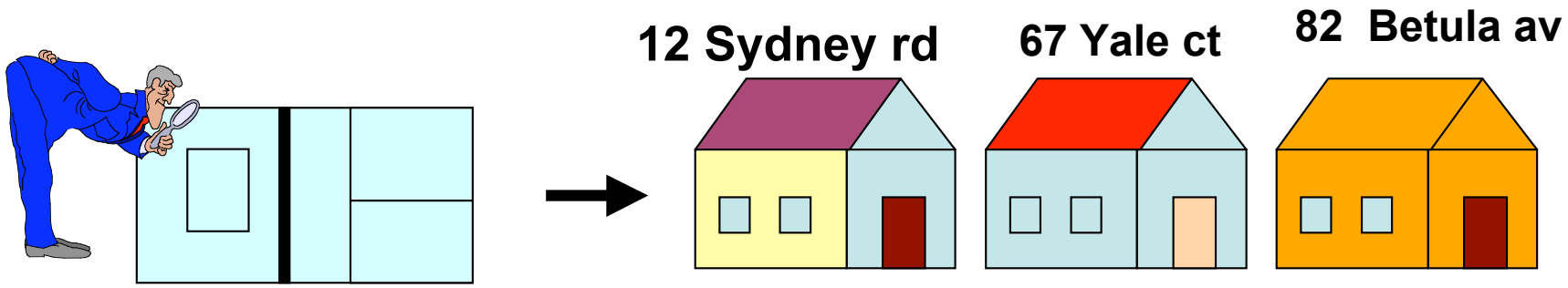


- The effect is still the same with two statements in the same line as in: (but considered poor style !)

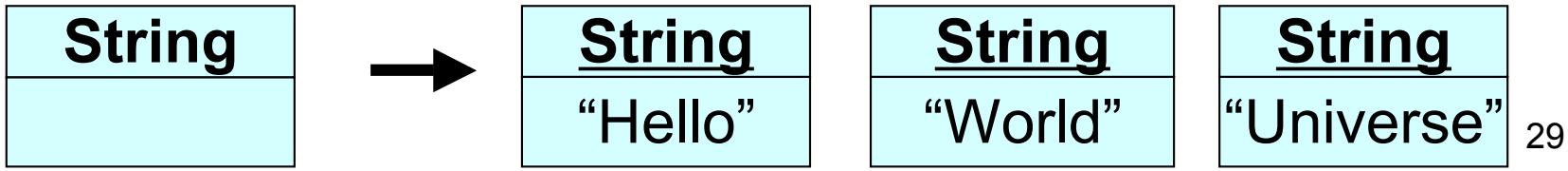
```
System.out.println(…); System.out.println(…);
```

Class and Objects

- **A class can be compared to a blueprint created by an architect when designing a house.**
- **It defines the important characteristics of the house such as walls, windows, electrical outlets etc.**
- **Once we have the blueprint several houses can be built using that . They may have different addresses, furniture, colors etc.**



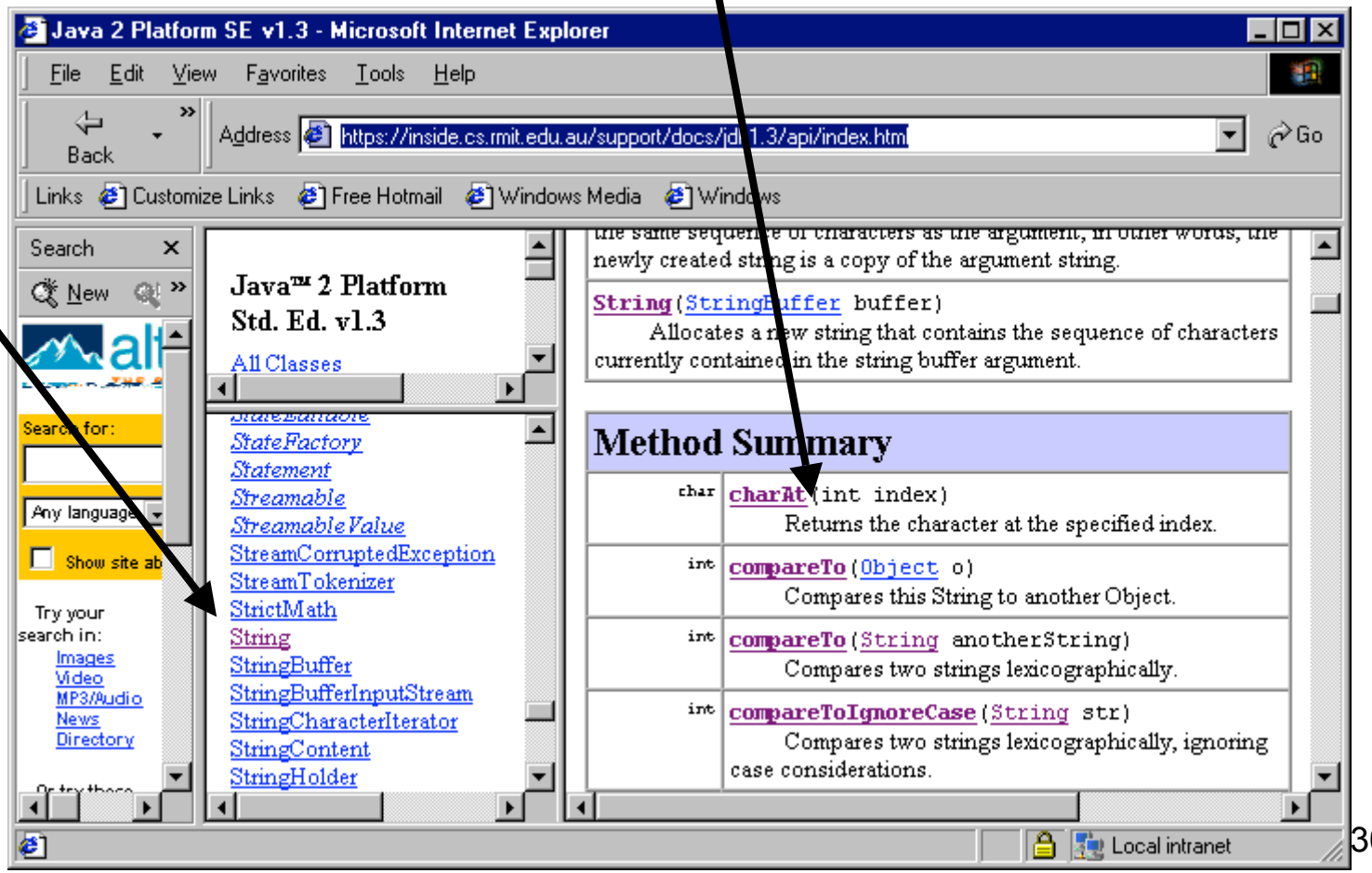
Similarly a class is a blueprint for objects.



Using the Java Docs

- **All the classes available and their method details can be viewed easily using the Java Docs**
- **Get lab-assistant's help this week**

classes



Creating new Objects with new

- In Java we are provided with a number of pre-created classes such as String, Rectangle ...
- New String and Rectangle objects can be created with these class by using the operator new (see below).
- The statement below creates a new Rectangle object and passes it to the println() method which is a method of the pre-defined PrintStream class.

```
System.out.println(new Rectangle(10,5,20,30));
```

- This method causes the details of the Rectangle object to be printed in the terminal.

```
java.awt.Rectangle[x=10,y=5,width=20,height=30]
```

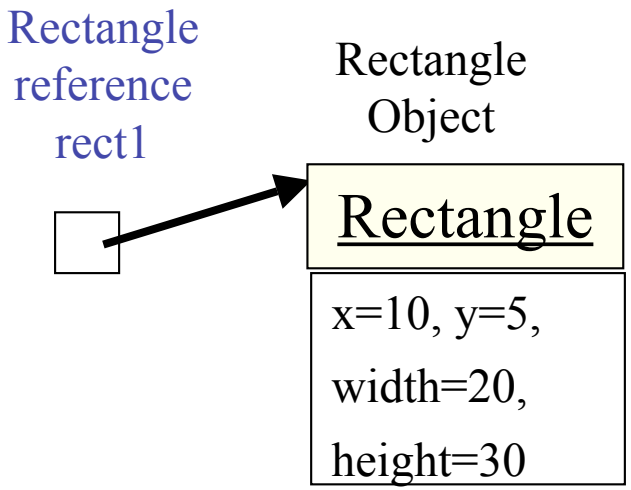
Getting a handle to an Object

- Suppose we want to manipulate the Rectangle object created, before printing its details (using one of its own methods).
- For example, we may want to apply the translate(int x, int y) method of Rectangle class on that object.
- Before we can call one if it's own methods, somehow we need to get a handle to the object !

- Next program declares a Rectangle reference and sets it to point (refer) to the newly created Rectangle object with:

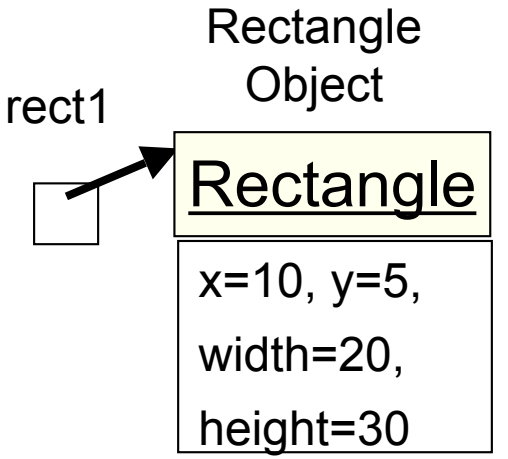
```
Rectangle rect1 = new Rectangle(10,5,20,30);
```

- Subsequently its own methods are called through that reference.



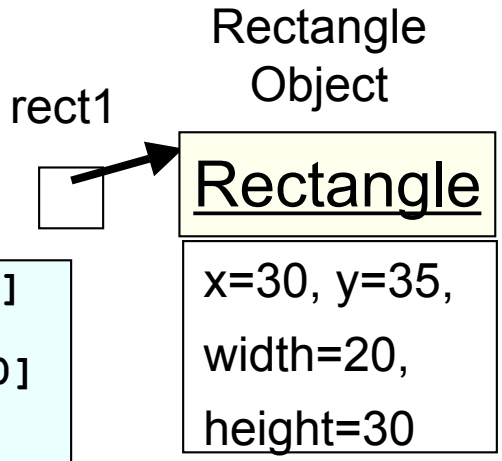
Manipulating the Rectangle Object

```
import java.awt.Rectangle;
public class HelloRectangle  {
   public static void main(String[] args){
      Rectangle rect1 = new Rectangle(10,5,20,30)
      System.out.println(rect1);
      rect1.translate(20,30);//shifting the location
     System.out.println(rect1);
   }
}
```



Output from the program

```
java.awt.Rectangle[x=10,y=5,width=20,height=30]
java.awt.Rectangle[x=30,y=35,width=20,height=30]
```



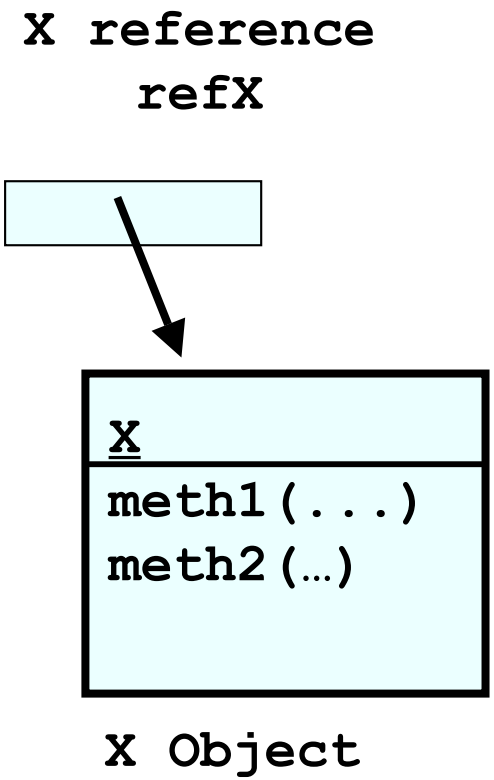
Creating and manipulating Objects Summary

- To access an object of class X say, we must declare a reference to that class X. Then this reference must be set to refer to that object with a statement of the form

```
X  refX  =  new  X(…);
```

- Now we can easily manipulate the X object using one of its own methods say meth1() as in

```
refX.meth1(...);
```



Time to pause….

1. Which of the following is true?

I. A class can have many objects (or instances)

II. An object (or instance) can have many classes

2. How do we create an object (or instance) of a specific class?

_____

Another Quiz

What will be the output of the program below?

```
public class RefTest
{
   public static void main(String args[])
   {
      String s;   // a String reference
      s = new String("Apple");
      s = new String("Orange");
      s = new String("Banana");
      System.out.println("s is now referring to " + s);
   }
}
```

Conclusion:

At any one time, a reference can refer to _____ (single/multiple) object(s).

Spot the Errors and state the Reasons

```
Rectangle r1 = new Rectangle(10,10,5,10); // A
String s = new String(" Good day ");      // B
String r2 = new Rectangle(10,10,5,10);    // C
String r3 = new String(" Good day ");     // D
```

Ans:

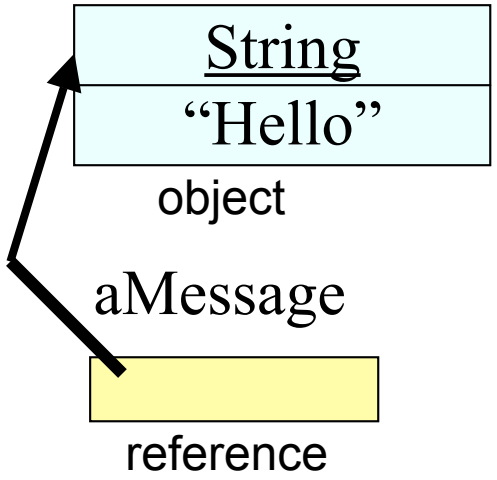
```
Rectangle r1 = new Rectangle(10,10,5,10);   // A
r1.translate(20,30);                        // B
String s1 = new String(" Monday ");         // C
s1.translate(20,30);                        // D
```

Ans:

String class - a special one

- program below declares a String reference aMessage and sets it to refer to a newly created String object.
- Subsequently we output that String object using the String reference aMessage .

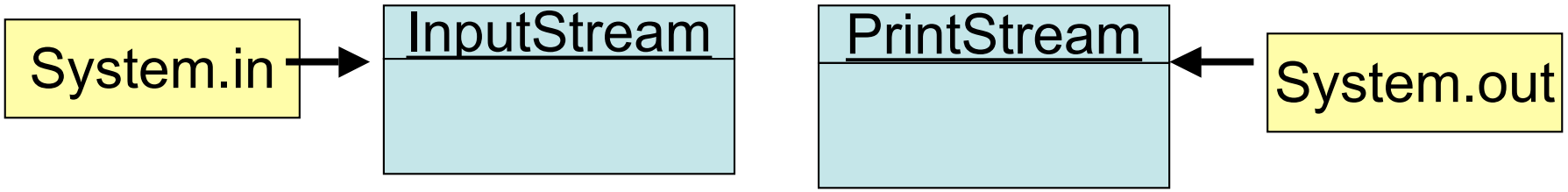
```
public class HelloWorld {
   public static void main(String[] args)  {
      String aMessage = new String("Hello");
      System.out.println(aMessage);
   }
}
```



- As String objects are commonly used, they need not be created explicitly using the operator new.
- Hence the 3rd line can be replaced with:
  `String aMessage =  “Hello";`

Java provides pre-created objects for Input/Output

- System.out refers to an object of PrintStream class.
- As console input/output is common in Java, they have pre-created objects of type PrintStream (for output) and InputStream (for input).
- To make them easily accessible the System classes contains references to these objects as shown here.



Pre-created InputStream Object (associated with keyboard)

Pre-created PrintStream Object (associated with terminal)

So what do I need to know about these precreated objects ?

- You are free to write to the terminal by writing directly to the object System.out.
- You are free to use any of the methods of PrintStream class
- Keyboard input requires little more processing - we will have to wait until next week

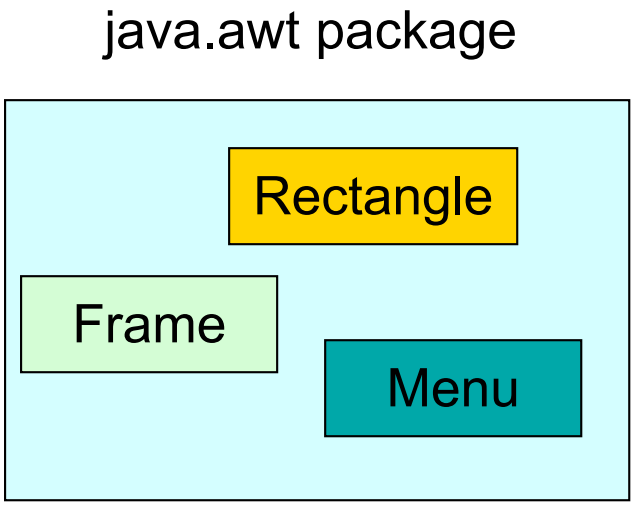
Using packages

- In the last program we specified that we are using the Rectangle class of the java.awt package.

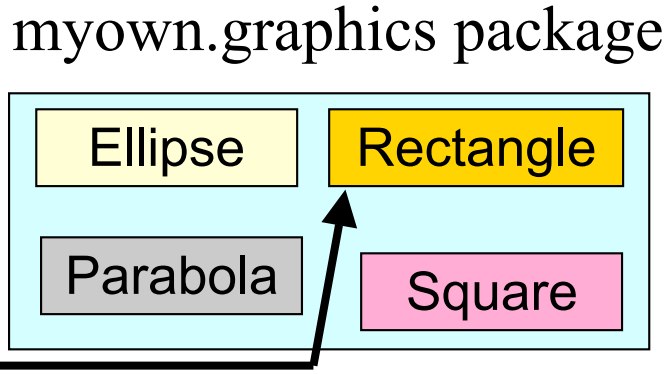
```
import java.awt.Rectangle;
```

- All classes in standard library are placed in packages: java.awt, java.io,
- To import all classes use:

```
import java.awt.*;
```



- Package java.lang is automatically imported - contains String,System, ...
- You can place your own classes in packages too - to avoid name clash.



Stages in Software Development

Specify

What must it accomplish ?

Analyze

Refine the requirements.

Design

How to do it ?

Use Algorithms, classes, Object Interactions

Implement

Code the programs

Test

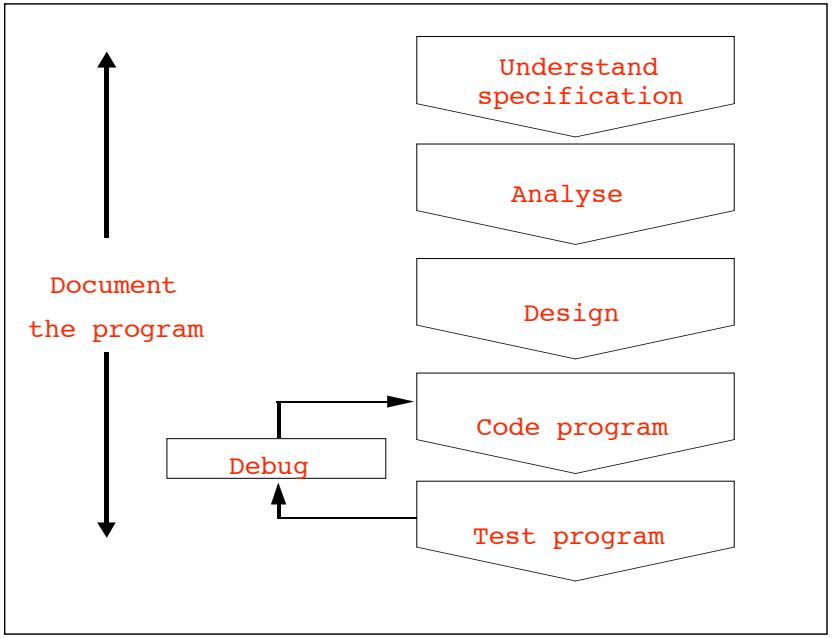
Run program with different inputs and verify results

Document

Make it maintainable

Write a program to find the roots (real) of a quadratic equation of the form:

$ax^2 + bx + c = 0.$



An Algorithm to find the roots of a quadratic equation

Algorithm is a step by step procedure in some form of English

1. **Prompt and read the coefficients a, b and c**
2. **Compute the roots if any and display**

Refine Step 2

2.1 Compute discriminant disc=b^2-4ac

2.2 If (disc>=0)

compute r1 = $\dfrac{-b + \sqrt{disc}}{2a}$

compute r1 = $\dfrac{-b - \sqrt{disc}}{2a}$

display r1 and r2

2.3 Otherwise display no Real Roots

Some Reserved Words

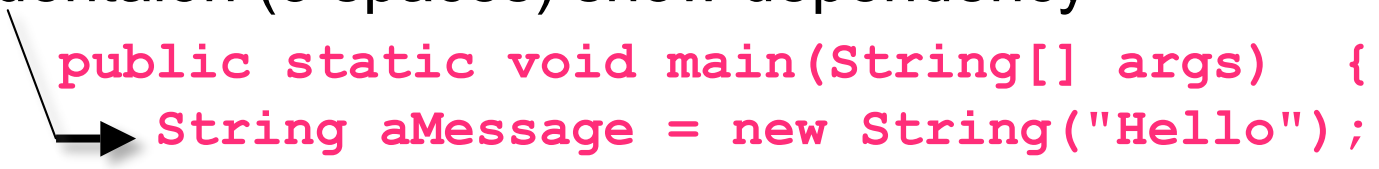
- Examples: class, static, public, void.
- They have been reserved by the designers and they can't be used with any other meaning.
- For a complete list see the prescribed book.

Programming style

- Java does not impose a particular style but there are rules most Java programmers follow.
- One statement per line

```
System.out.println("Welcome to RMIT");
System.out.println("Welcome to CS");
```

- Indentaion (3 spaces) show dependency



```
public static void main(String[] args)  {
   String aMessage = new String("Hello");
```

- Comments help clarify the code
- Use blank lines to separate logical sections
- Use meaningful identifiers but not verbose

Finding the roots of a quadratic Equation

```
/* Written by CT 23/02/2006
   This program finds the roots of a quadratic equation
*/

import java.util.*;
public class FindRoots {
   public static void main (String[] args) {
      double a,b,c;               // coefficients of the equation
      double r1=0.0, r2 = 0.0;    // real roots of the equation
      double disc;                // discriminant
      Scanner console = new Scanner(System.in);

        // Getting user input for coefficients
      System.out.println ("Enter value for a");
      a = console.nextDouble();
      System.out.println ("Enter value for b");
      b = console.nextDouble();
      System.out.println ("Enter value for c");
      c = console.nextDouble();
```

Finding the roots of a quadratic Equation

```
// Computing results
      disc = b*b - 4*a*c;
      // no real roots exist if discriminant is negative
      if (disc > 0.0) {
           r1 = ( -b + Math.sqrt(disc)) / (2*a);
           r2 = ( -b - Math.sqrt(disc)) / (2*a);
      }

       // Displaying results
      if (disc >= 0)
           System.out.println("Roots are " + r1+ " and " +r2);
      else  System.out.println("No real roots exist");
   }
}
```

Sample User Interface for FindRoots program

```
Enter the value for a
1.0
Enter the value for b
3.0
Enter the value for c
2.0
The roots are -1.0 and -2.0
```

```
Enter the value for a
4.0
Enter the value for b
2.0
Enter the value for c
6.0
No real roots exist
```

What if I input an alphabet such as ‘s’ as input to a ?

Choosing Identifiers

- Identifiers should be meaningful, but not verbose.
- Any combination of letters, digits, dollar signs '$' underscore characters '_', not beginning with a digit, is legal.
- Legal: Total, lastWord, TaxCalculation
- Illegal: 3rdAmmendment, you too, you#too
- Avoid: s1, theFirstOfTheStudentsInTheClass
- Acceptable : student1, stud_1, firstStudent

Programming Errors: Compilation Errors

- detected at compile time (javac hello.java)
- caused by undeclared variables, missing semicolons, incompatible types

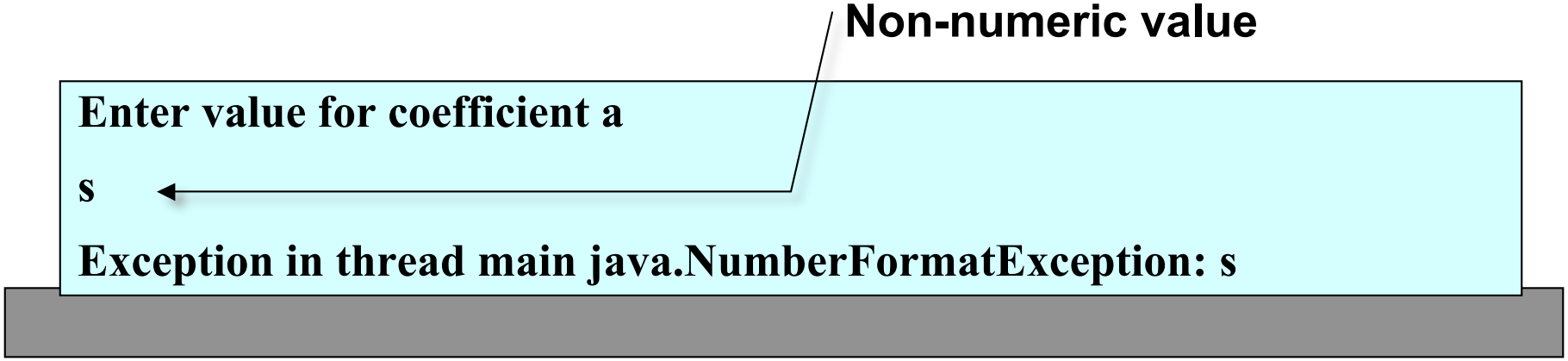
```
disc = b*b - 4*a*c    // missing semicolon
```

```
double r1 =0.0, r2=0.0;
….
root1=(-b + Math.sqrt(disc))/(2*a);  // undeclared variable root1
```

```
int disc;
..
disc = b*b-4*a*c;  // double value cannot be assigned to int
```

Programming Errors: Execution Errors

- appear when the program runs (java Hello).
- Examples are, a division by zero, an input of the wrong type, finding the square root of a negative number etc.
- Typically execution stops when an exception such as these happens.
- For example if in the FindRoots program we input an alphabet (such as s) execution will stop with an error.



Programming Errors: Logic Errors

- program compiles and runs, but the results are not what they should be.
- For example in the FindRoots program if we write

```
r1 =  -b + Math.sqrt(disc) / (2*a);
```

instead of

```
r1 =  (-b + Math.sqrt(disc)) / (2*a);
```

the wrong roots will be printed.

Exercise

A. The program below written to find average of two weights has 3 syntax (compilation) errors and 1 logical error. Identify and correct them.

```
public class Add
{
   public static void main(String args[])
   {
       double weight1 = 85.5
       double weight2 = 92.3

       double averWeight = weight1 + weight2 / 2;
       System.out.println("Average is " + aver);
   }
}
```

B. What changes are needed to find average of 3 weights 85.5,92.3 and 78.4 ?_____

C. What changes are needed to find average of other weights without having to change the program ? _____